EFFICIENT GEAR-SHIFTING FOR A POWER-PROPORTIONAL DISTRIBUTED DATA-PLACEMENT METHOD

2014/1/27

Hieu Hanh Le, Satoshi Hikida and Haruo Yokota

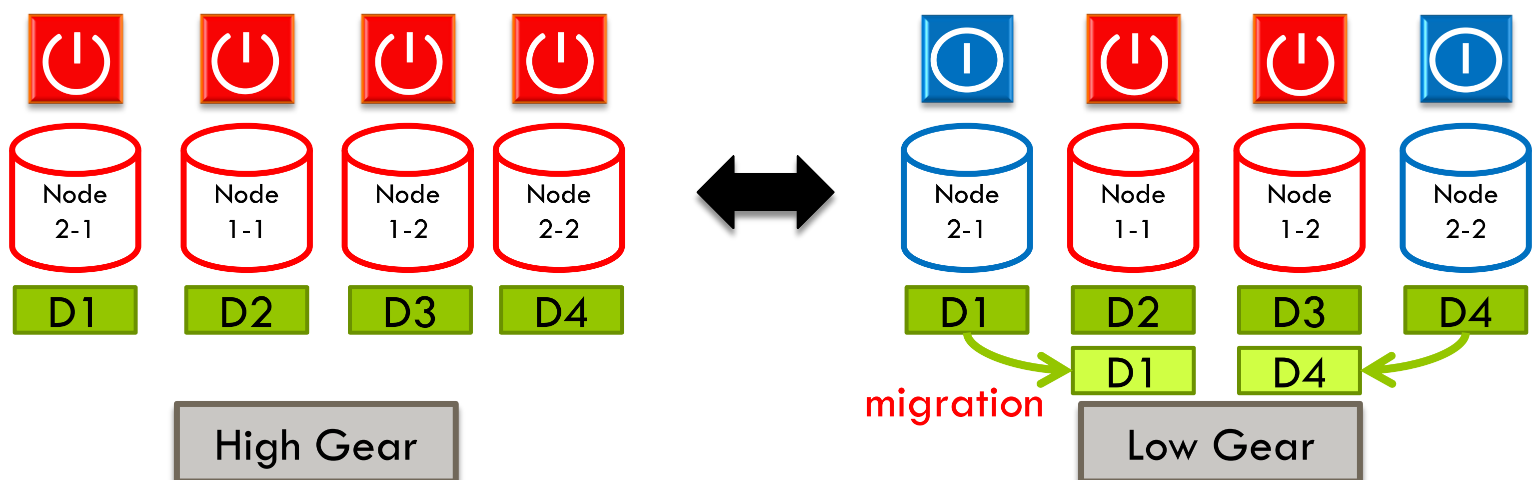
Tokyo Institute of Technology

1.1 Background

- Commodity-based distributed file systems are useful for efficient Big Data processing
  - Hadoop Distributed File System (HDFS), Google File System
  - Support MapReduce framework
  - Gain good scalability
    - Utilize a large number of nodes to store huge amount of data requested by data-intensive applications
      - Also expand the power consumption of the storage system
- Power-aware file systems are increasingly moving towards power-proportional design

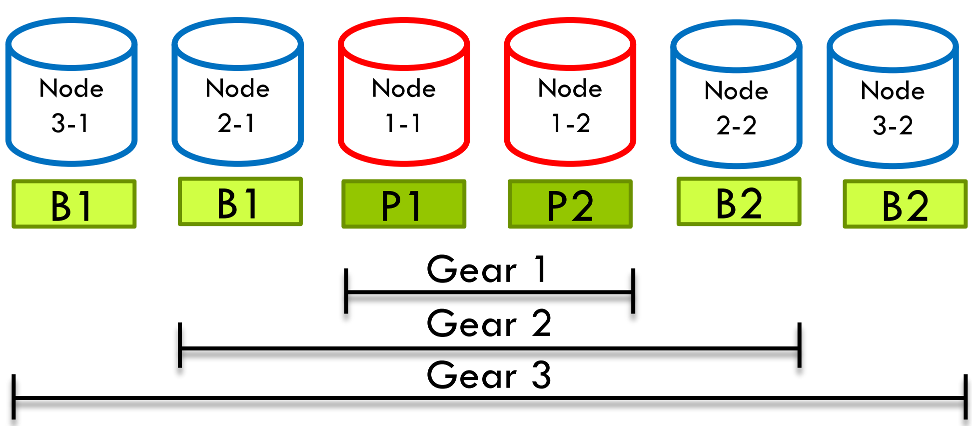
1.2 Power-proportional Storage System

- System should consume energy in proportion to amount of work performed [Barroso and Holzle, 2007]
- Set system's operation to multiple gears containing different number of data nodes
- Made possible by data placement methods



2.1 Current Data Placement Methods

- Rabbit [ACM SOCC 2010], Sierra[ACM EuroSys 2011]
  - The primary of the dataset are stored entirely in the group of Covering-set nodes which are always active
  - The backups of the dataset are stored entirely in each group of nonCovering-set nodes which can be inactive



Group	Nodes
Group 1	Node 1-1, Node 1-2
Group 2	Node 2-1, Node 2-2
Group 3	Node 3-1, Node 3-2

P: Primary data
B: Backup data

2.2 Motivation

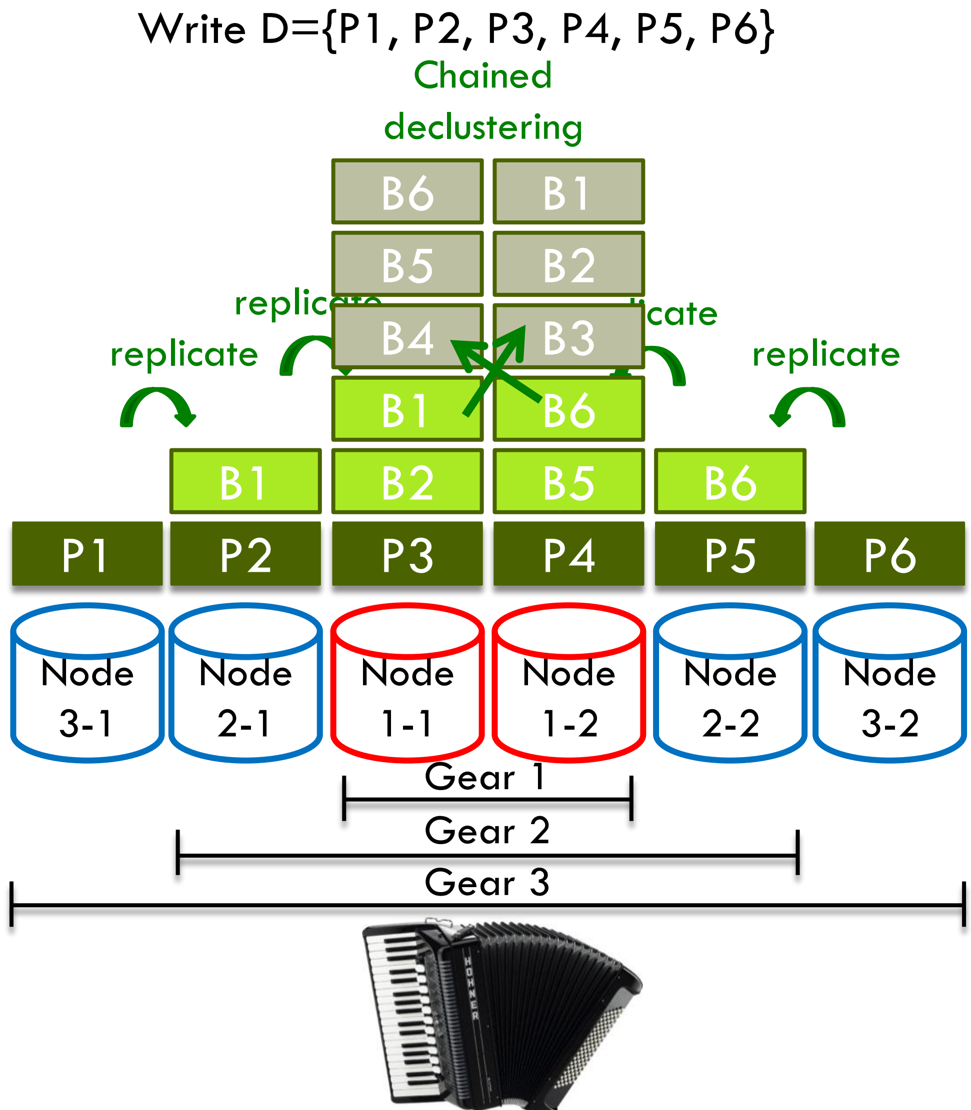
- Efficient gear-shifting becomes vital in power-proportional system
  - Deal with write requests when operating in a low gear
    - Apply Write-Offloading [ACM Trans. On Storage, 2008]
    - The data that should be written to deactivate nodes are temporally stored at activate nodes
    - When the system shifts to a high gear, all the temporal data have to be transferred to the reactivated nodes according to policy
  - Reduce performance degradation when shifts to higher gear
    - Still an open issue

3. Goal and Approach

- Propose Accordion
  - A power-proportional data placement method with efficient gear-shifting
- Approach
  - Control the power consumption of the system by dividing the nodes into separate groups
  - Carefully design the data replication strategy to provide efficient gear-shifting
    - Location of the primary data becomes the key to reduce the temporal data needed to be retransferred

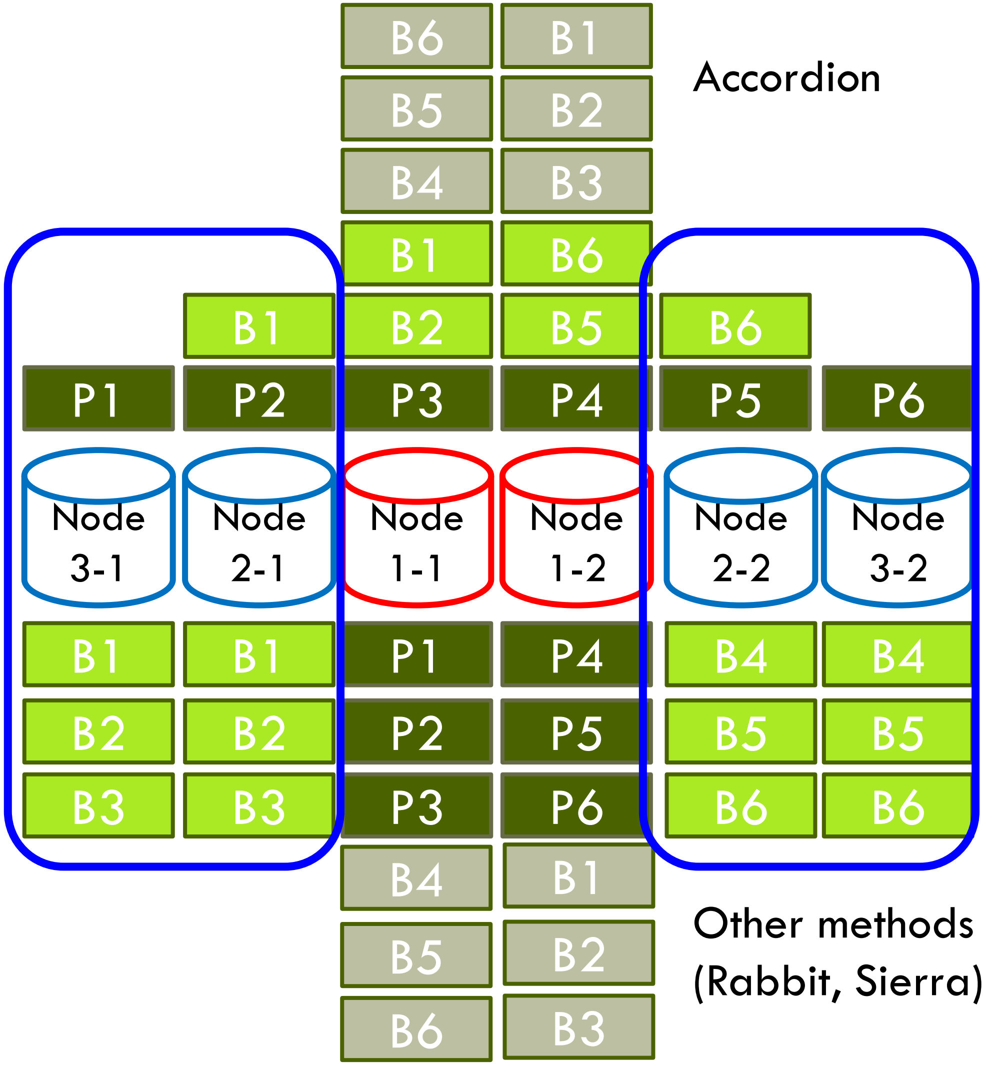
4.1Data Replication Strategy in Accordion

- Primary data
  - Evenly distribute primary data to all of the nodes in the systems
- Backup data
  - Replicate the data from outbound nodes to inbound nodes step-by-step
  - Finally, apply the chained declustering policy at Convering-set nodes
    - Enhance reliability at the lowest gear

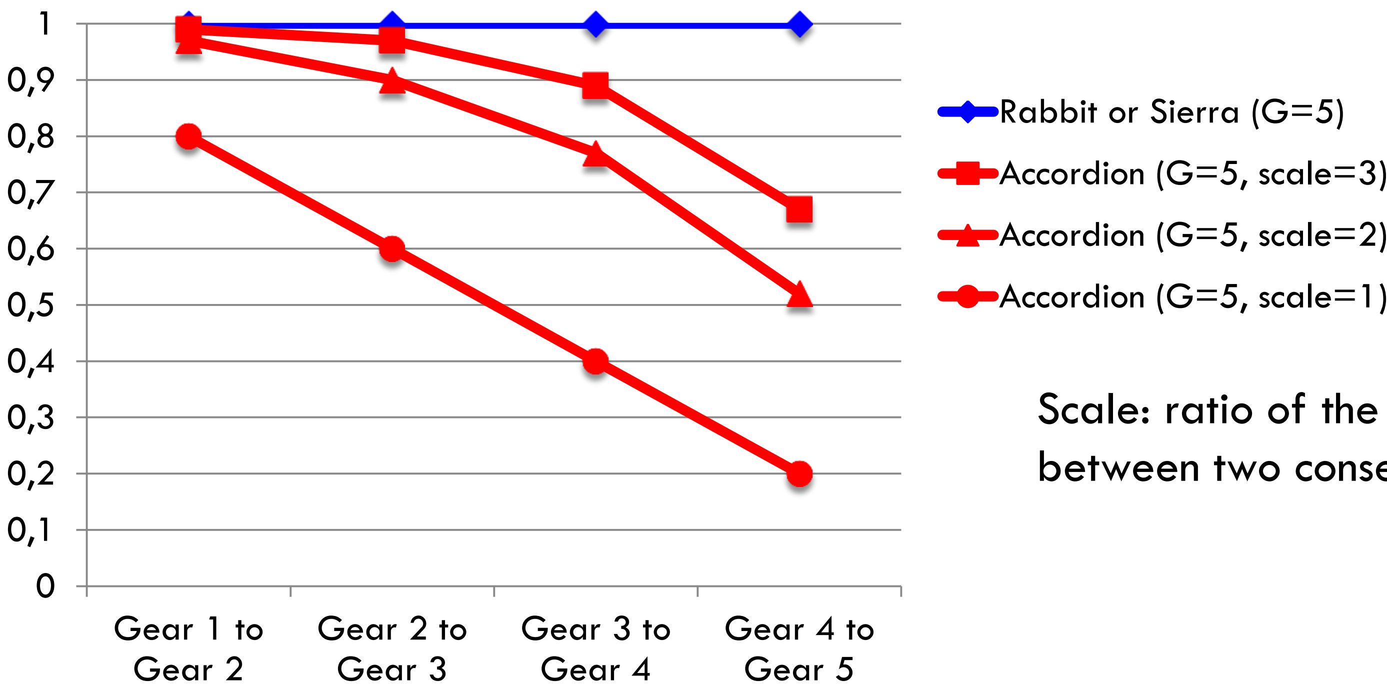


4.2 Accordion vs. Other Methods

- The amount of retransferred data is smaller than in other methods
  - The retransferred data are the data that should be written to non-CoveringSet nodes



4.3 Accordion in Multiple-Gear File System



Scale: ratio of the number of nodes between two consecutive groups

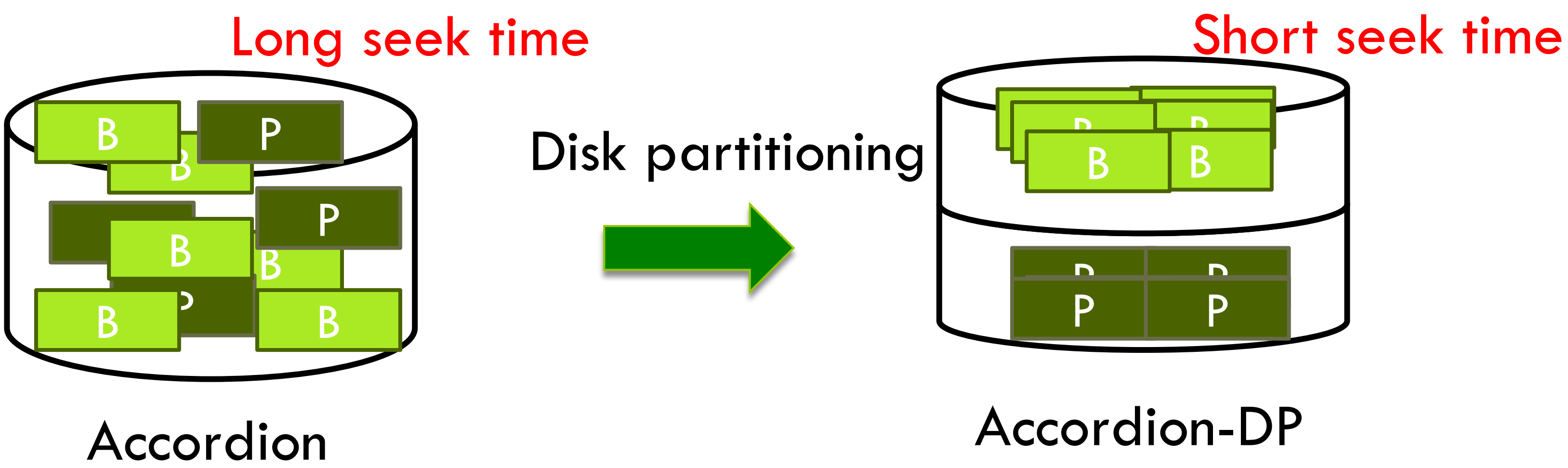
- The amount of retransferred data becomes smaller when the system perform gear-shifting in a higher gear
  - The number of inactive nodes is decreasing

4.4 The Skew in Data Distribution in Accordion

- An imbalance in the amount of data does not lead to a considerable problem
  - Current load balancing mechanisms only make use of part of the capacity other than full of the capacity
  - The load can be well distributed to all the nodes
    - The same amount of data are served by each node
    - Highly take advantage of parallelism in serving the request

4.5 Physical Data Placement on an Accordion Node

- Deal with the mixture of primary and backup data
  - Majorly only primary or backup data are accessed
- **Accordion**-with-**D**isk-**P**artition (Accordion-DP)
  - Utilize partitioning technique to separate the data
  - Improve the performance through reducing the seek time on disks



5. Empirical Experiments

- Develop a prototype of Accordion based on a modified HDFS which is able to perform:
  - Different data placement methods
  - Load balancing
  - Writing requests in a low gear based on Write-Offloading
- Extensive experiments
  - Evaluate the effectiveness of load balancing
  - Verify the effectiveness of Accordion on efficient gear-shifting
  - Evaluate the power-proportionality of Accordion under different system configurations

5.1 Experiment Framework

Low power nodes

	Node
Name	ASUS Eeebox EB100
CPU	TM8600 1.0GHz
Memory	DRAM 4GB
NIC	1000Mb/s
OS	Linux 3.0 64bit
Java	JDK-1.7.0

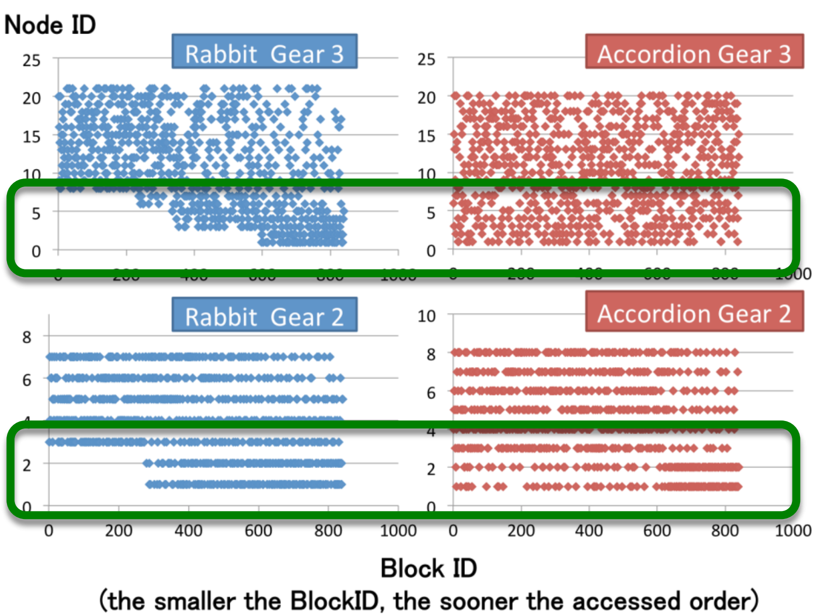
Power consumption measurement HIOKI 3334

5.2 Effect of Load Balancing

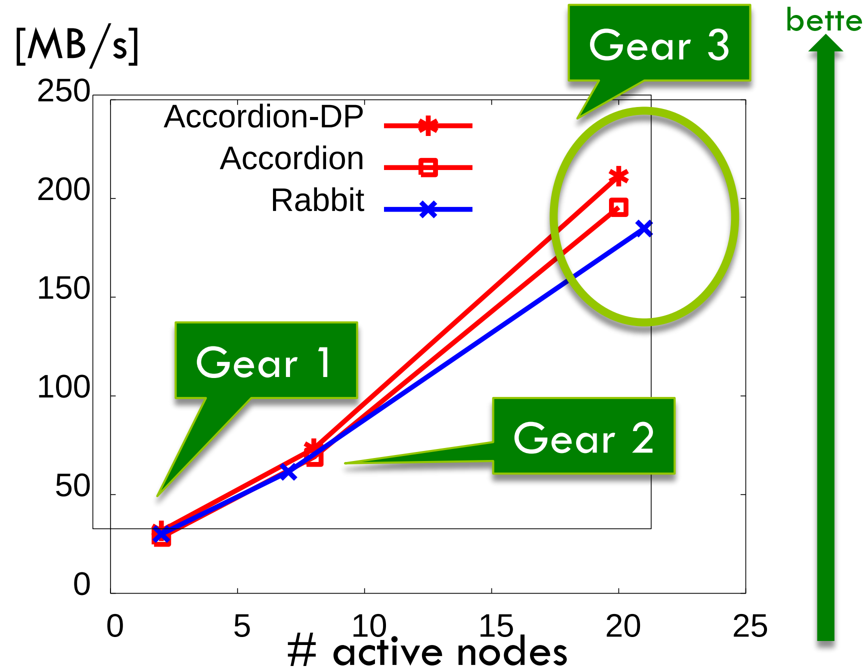
- Evaluate the throughput performance
  - Methods: Accordion, Accordion-DP, Rabbit
  - Throughput of the workload of reading a dataset
- Experiment environment

# Gears	3
# Active nodes (Accordion, Accordion-DP)	2, 8, 20
# Active nodes (Rabbit)	2, 7, 21
# Files	420
File size	64 MB
HDFS version	0.20.2
Block size	32 MB

5.2.1 Effect of Load Balancing



Distribution of accessed time



Average throughput

- Accordion-based methods gained better throughput, especially in Gear 3 due to better distribution of serving time
- The effective of Disk-partitioning was verified as Accordion-DP achieved better result than Accordion

5.3 Efficient Gear Shifting

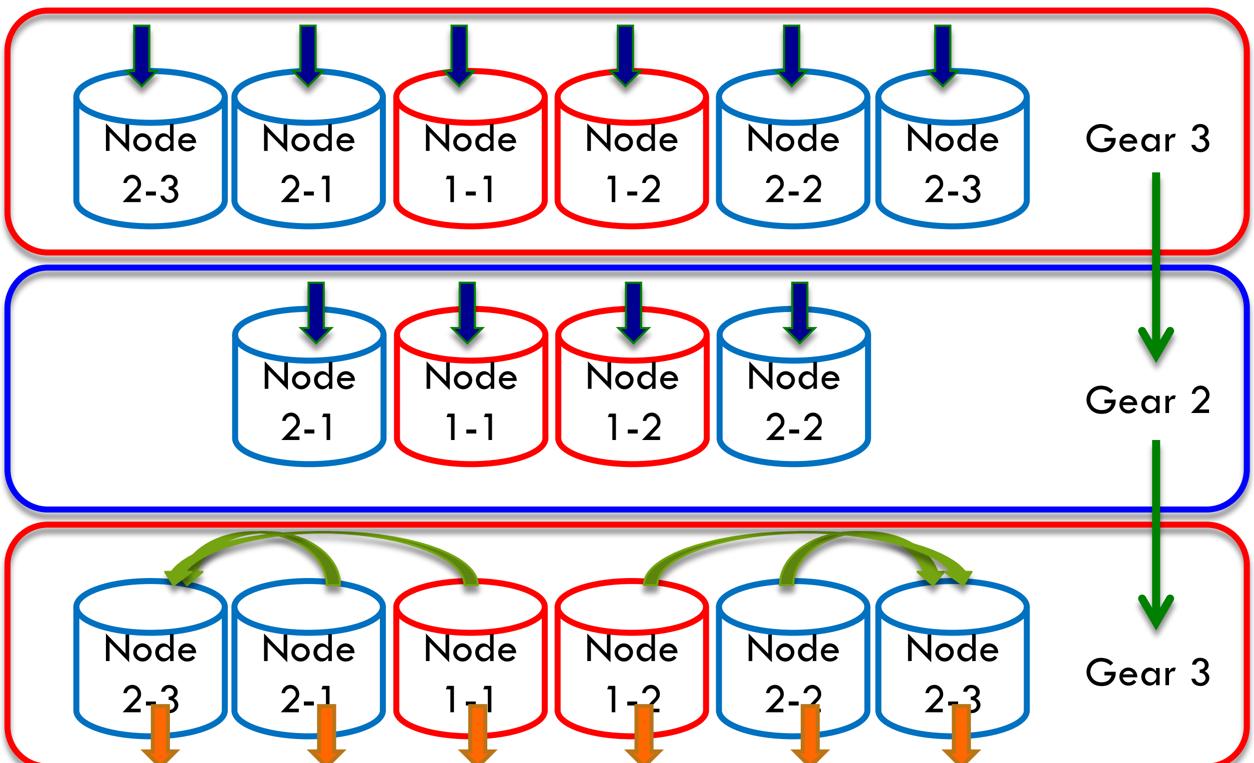
- Gear shifting scenario

1. Write a part of dataset (W1 files)

2. Write a part of dataset (W2 files) with Write-Offloading

3.1 Retransfer data

3.2 Read a whole dataset (W1+W2 files)

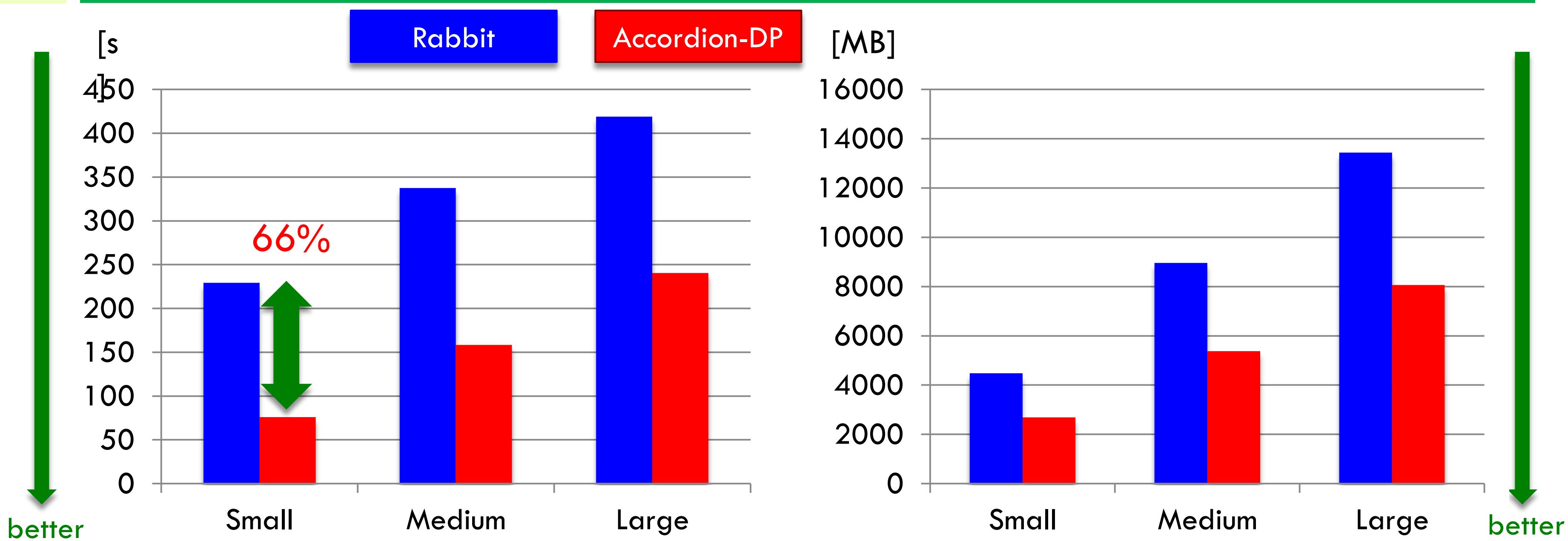


5.3.1Experiment Method

- Evaluate the performance when the system shifts to a higher gear
  - Execution time for retransferring data
  - Effect of retransferring data to the performance
    - Accordion-DP and Rabbit
    - 4 configurations with the change in the ratio of W1 and W2

Configuration	W1 (files)	W2 (files)
Without Updated Data	420	0
Small	350	70
Medium	280	140
Large	210	210

5.3.2 Data Retransferred Process

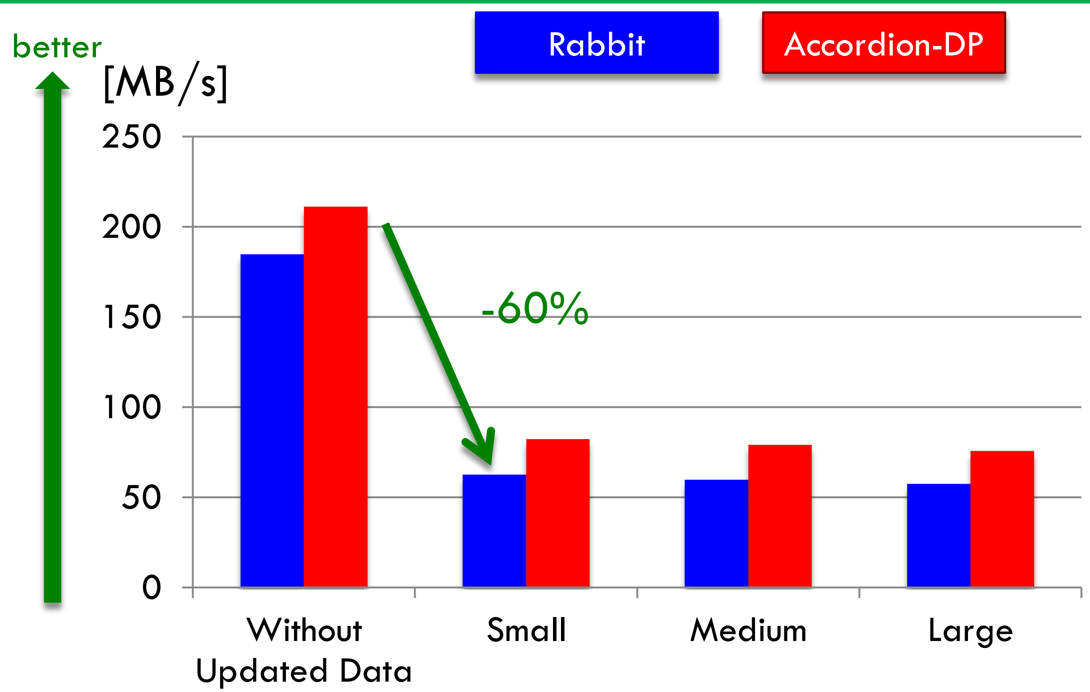


The execution time for retransferring temporal data

The size of retransferred data

- Accordion-DP significantly reduced the execution time by up to 66%
- The data size of retransferred data in Accordion-DP was always smaller than in Rabbit

5.3.3 Throughput of Reading Whole Dataset



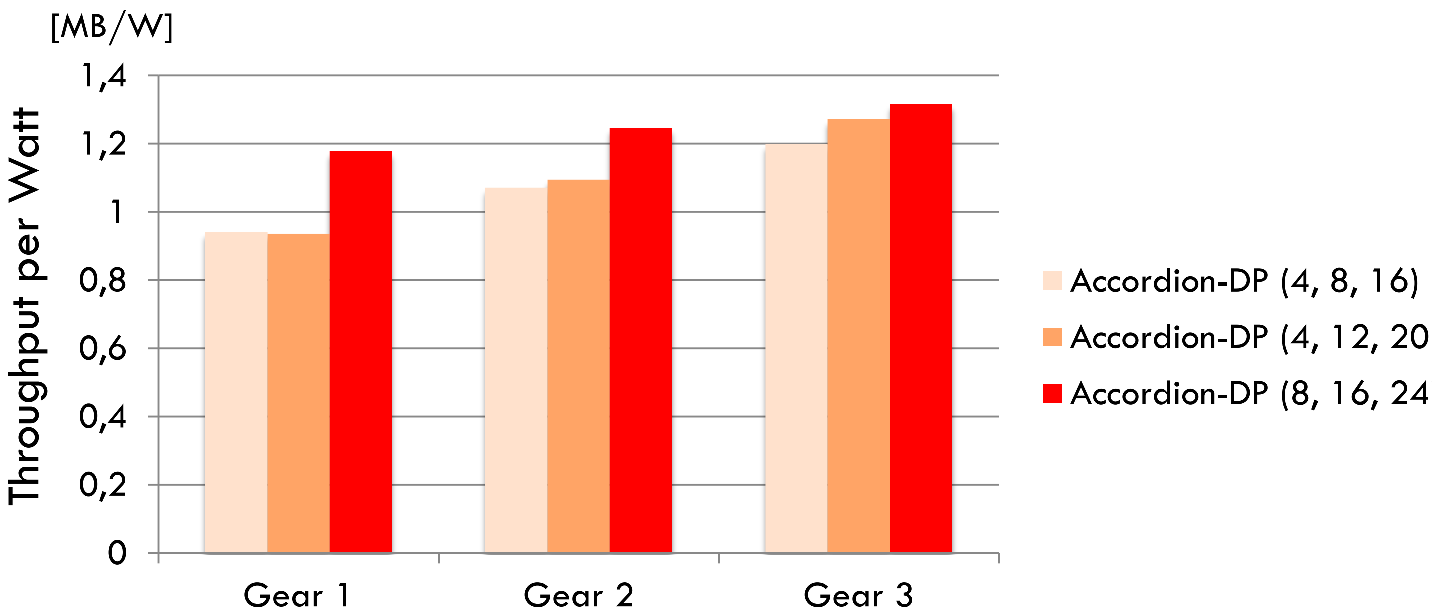
- Throughput was significantly degraded due to data retransferred process by more than 60%
- Accordion-DP always gained better throughput by 30%

5.4 Effect of System Configuration

- Evaluate the effects of different configuration of Accordion on the power-proportionality
  - 3 gears with 3 configurations
  - The throughput of reading 420 files workload
  - The power consumption of storing nodes

Configuration	#active nodes Gear 1	#active nodes Gear 2	#active nodes Gear 3
Accordion-DP (4, 8, 12)	4	8	12
Accordion-DP (4, 12, 20)	4	12	20
Accordion-DP (8, 16, 24)	8	16	24

5.4.1 Effect of System Configurations on Performance



- Largest-scale configuration yielded best result due to the load at each node is the lightest
- The higher gears gained better results in case of the same active nodes
  - The less complexity of primary and backup data

6.1 Conclusion

- Propose an Accordion data placement for efficient gear-shifting
  - Reduced the execution time required for data movement by 66% and improved the performance by 30% compared with Rabbit
  - Ensured the smaller amount of reallocated data and achieve better power-proportionality
    - Different in primary data location
    - Utilize the partitioning technique to reduce the seek time at each node
  - Shown the high potential for deployment in large-scale systems

6.2 Future Work

- More experiment with different workloads and system configurations
  - Larger the scale, larger the number of gears
- Improve the load balancing algorithm
- Integrate Accordion with other architecture than HDFS
- Consider a specific algorithm to deal with system failures

THANK YOU FOR YOUR ATTENTION

Q&A